UNITED STATES PATENT APPLICATION
FOR

ADAPTIVE QUEUE SCHEDULING

INVENTOR:

ADDICAM V. SANJAY,
a citizen of the United States of America

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030
(303) 740-1980

EXPRESS MAIL CERTIFICATE OF MAILING
"Express Mail" Mailing No. EV 331619565 US

I hereby certify that I am causing the above-referenced correspondence to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated below and that this paper or fee has been addressed to Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.
Date of Deposit: December 30, 2003
Name of Person Mailing Correspondence: Krista Mathieson

Krista Mathieson — December 30, 2003
Signature Date

ADAPTIVE QUEUE SCHEDULING

BACKGROUND

[0001] In computer operations, scheduling of tasks can greatly affect system performance. For example, if multiple tasks are pending and such tasks have varying priorities, then scheduling is intended to attend to each of the multiple tasks while also addressing the priorities of the tasks.

[0002] Scheduling may be implemented in many different ways. In one example, round robin scheduling provides that tasks are ordered according to their priorities and then sent to a device driver in groups or sets based on such priorities. The groups or sets may contain more of the higher priority packets, providing for weighted round robin scheduling.

[0003] However, conventional scheduling algorithms may schedule certain tasks ahead of higher priority tasks because of the structure of the operation. In conventional round robin scheduling, weighted round robin scheduling, or similar processes packets may continue to be scheduled from a queue after there are no more high priority packets remaining in the queue. Because of this, lower priority packets may be scheduled even though higher priority packets may have been received.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] The invention may be best understood by referring to the following description and accompanying drawings that are used to illustrate embodiments of the invention. In the drawings:

[0005] **Figure 1** illustrates an embodiment of scheduling prioritization;

[0006] **Figure 2** illustrates an embodiment of queue formation for scheduling;

[0007] **Figure 3** illustrates an embodiment of round robin scheduling by a scheduler;

[0008] **Figure 4** is a flow chart of an embodiment of adaptive queue scheduling;

[0009] **Figure 5** is a flow chart of an embodiment of adaptive queue scheduling utilizing a timer; and

[0010] **Figure 6** illustrates an embodiment of a computer environment.

DETAILED DESCRIPTION

[0011] A method and apparatus are described for adaptive queue scheduling.

[0012] Before describing an exemplary environment in which various embodiments of the present invention may be implemented, certain terms that will be used in this application will be briefly defined:

[0013] As used herein, "packet" means a unit or set of data. A packet may, for example, represent a task, instruction, or set of instructions.

[0014] As used herein, "round robin scheduling" means a scheduling algorithm or method in which processes are scheduled in a circular or alternating pattern from various sources. The term includes scheduling in which tasks are scheduled alternately from multiple different priority levels.

[0015] As used herein, "weighted round robin scheduling" means round robin scheduling in which the scheduling of processes is weighted on some basis. The term includes round robin scheduling in which the number of tasks scheduled is weighted based upon the priority of each task.

[0016] Under an embodiment of the invention, adaptive queue scheduling is provided. In one embodiment, a scheduling algorithm includes scheduling tasks from various sources or queues. If there are no more tasks available from one or more sources or queues, the scheduling is paused, additional tasks are obtained, and the scheduling is then resumed.

[0017] According to one embodiment of the invention, a system utilizes round robin scheduling or a similar scheduling algorithm. The algorithm schedules tasks from multiple queues (or sub-queues within a queue), with each queue representing a particular

priority level or levels. If it is determined that there are no more tasks in the highest priority queue, the algorithm pauses the scheduling process to obtain more tasks for scheduling and then resumes scheduling. A particular embodiment includes a delay process. In this embodiment, if a process obtains more tasks for scheduling and determines that there are no tasks in the highest priority queue, then a delay period is commenced. During the delay period, the process does not obtain more tasks when it is determined that there are no tasks in the highest priority queue. The delay period may provide a time period in which scheduling is continues while time is allowed for more high priority tasks to arrive.

[0018] The examples provided herein show a determination whether there are any tasks with the highest priority level. Embodiments of the invention are not limited to these examples. Embodiments may also apply when a determination is made whether any tasks in a number of different priority levels, such as the two highest priority levels. Embodiments of the invention are not limited to processes in which tasks are assigned specific priority levels, and may be applied in any situation in which tasks have discernable characteristics or are received from different sources.

[0019] **Figure 1** illustrates an embodiment of scheduling prioritization. In this illustration, a scheduler **105** is responsible for scheduling tasks to be provided to a unit, such as driver **120**. The tasks are represented by packets of data. Received packets **110** are of varying priority levels. In the illustrated example, each packet has one of three different priority levels, but embodiments of the invention are not limited to any particular number or type of priority levels. As shown, a first, highest priority level is

designated as P1. A second priority level that is of lower priority than P1 is P2. A third priority level that is of lower priority than P2 is P3.

[0020] The received packets **110** are scheduled by the scheduler utilizing the priority levels of the packets. In this illustration, a set of packets **115** is scheduled comprising three of the P1 priority packets, two of the P2 priority packets, and one of the P3 priority packets. The scheduling of the packets represents a weighted round robin scheduling in which sets of packets are scheduled, with the number of packets in each set being weighted according to the priorities of the packets. In scheduling, the scheduler **105** may run out of the highest priority packets (or out of one or more priority levels) in the packets being scheduled.

[0021] According to an embodiment of the invention, the scheduler **105** will proceed to obtain more packets when no more high priority packets, such as P1 packets, are available. In this embodiment, the scheduler **105** therefore may avoid scheduling lower priority packets when higher priority packets can be obtained for scheduling. According to a particular embodiment, scheduler determines whether any high priority packets are obtained after obtaining the additional packets and, if no high priority packets have been obtained, commences a delay period during which period the scheduler may continue to schedule the previously obtained packets without obtaining more packets.

[0022] **Figure 2** illustrates an embodiment of queue formation for scheduling. In this illustration, the processes are described in terms of a first thread (thread 1 **205**) that relates to the formation of queues for received packets into a priority packet queue **230** and a second thread (thread 2 **210**) that relates to copying packets from the priority packet queue into a copied packet queue **250**. Other embodiments may institute the

processes in a different manner. While this description refers to ordering packets and copying packets, this may be understood to mean that pointers or other similar devices are used to reference the packets. The priority packet queue **230** and the copied packet queue **250** each comprise multiple individual queues of packets, which may be referred to as sub-queues. In this example, each individual sub-queue represents one of a number of different priority levels. This structure may vary in different embodiments of the invention. For example, one sub-queue may represent more than one different priority levels.

[0023] In thread 1 **205**, a lock is obtained on the priority packet queue **215** to prevent any other access to the queue while the operation is underway. Received packets are then separated into sub-queues based on the priorities of each packet. In this example, the priorities comprise a priority 1 **235**, a priority 2 **240**, and continuing through a priority *n* **245**. Each priority queue may include a number of packets of such priority or may be empty if no packets of such priority are currently present. According to one embodiment of the invention, each sub-queue is organized according to the time such packet arrived or some other time factor. After the received packets have been organized into the priority packet queue **220**, thread 1 **205** provides for giving up the lock for the priority packet queue **275** to allow access by other operations. Thread 1 **205** continues in this pattern as packets are received.

[0024] In thread 2 **210**, a lock is obtained on the priority packet queue **215** to prevent any other access to the queue while the operation is underway. The thread continues with copying any uncopied packets **225** from the priority packet queue **230** to a copied packet queue **250**. As illustrated in Figure 2, the copied packet queue **250**

contains parallel priority sub-queues with the priority packet queue **230**, these being priority 1 **255**, priority 2 **260**, and continuing through priority *n* **265**. After copying any previously un-copied packets, thread 2 **210** provides for giving up the lock for the priority packet queue **275** and operating the scheduling algorithm module **280**.

[0025] **Figure 3** illustrates an embodiment of round robin scheduling by a scheduler. In this illustration, an algorithm **305** controls a scheduling operation for packets that are to be provided to, for example, a device driver **330**. The scheduling operation provides for scheduling of packets of multiple different priorities. The packets illustrated are in a priority 1 queue **310**, a priority 2 queue **315**, and a priority 3 queue **320**, although any number of priority levels and types may be present. To illustrate the round robin scheduling operation, a scheduling unit **325** takes one or more packets from priority 1 **310**, one or more packets from priority 2 **315**, and one or more packets from priority 3 **320**. The scheduling unit **325** then begins again with priority 1 **310**, as shown figuratively by the scheduling unit **325** rotating from each priority level to the next level.

[0026] In a weighted round robin scheduling operation, the number of packets scheduled for each priority is weighted based on the priorities of the packets. For example, three packets may be obtained from the priority 1 queue **310**, two packets may be obtained from the priority 2 queue **315**, and one packet may be obtained from the priority 1 queue **320**. The scheduled packets are provided to device driver **330**. In the process of scheduling packets, the packets in the highest priority queue, priority 1 queue **310**, may be exhausted. According to an embodiment of the invention, if the packets in the highest priority queue are exhausted, the algorithm **305** provides for copying additional packets before scheduling is resumed. In another embodiment, if additional

packets are copied and there are still no packets in the highest priority queue **310**, then a delay period may be commenced. The scheduling then will continue with the available packets without copying any more packets until the delay period has expired. While for simplicity this example only refers to exhausting the highest priority queue, other embodiments might provide for copying packets if one or more different priority queues are exhausted.

[0027] **Figure 4** is a flow chart of an embodiment of adaptive queue scheduling. In this illustration, received packets are copied into a copied packets queue for scheduling **405**. The copied packets queue includes packets arranged in a number of sub-queues, each sub-queue representing one of a number of different priorities. In this example, three priority levels (P1 - high, P2 - medium, P3 low) are provided, but any number and type of priority levels may be provided. To the degree that packets are available, a certain number of packets are scheduled from each priority level. In a weighted scheduling algorithm, the scheduling of the packets is weighted according to the priority levels of the packets. In this example, a driver **425** is sent a set of three P1 packets **410**, two P2 packets **415**, and one P3 packet **420** (if this number of packets is available for each priority level) in each scheduling action.

[0028] After a set of packets has been scheduled and sent, there is a determination whether all of the sub-queues are empty **435**. If so, then there are no packets available for scheduling, and the process continues with copying received packets into the copied packets queue **440**. The process then returns to sending packets from each of the priority sub-queues **410-420**. If all of the priority sub-queues are not empty **435**, there is then a determination whether the high priority sub-queue P1 is empty **445**.

If the P1 sub-queue is not empty, then the process of scheduling packets from the priority sub-queues continues **410-420**. However, if the P1 sub-queue is empty, then the process provides for copying received packets into the copied packets queue **440** and proceeding to the process of scheduling packets from the priority sub-queues continues **410-420**. By copying such received packets, the process may avoid scheduling lower priority packets while higher priority packets are not addressed. While the illustrated operations in Figures 4 and 5 make a determination whether any of the highest priority (P1) packets are available in the queue, embodiments of the invention are not limited to this type of example. An embodiment may, for example, make a determination whether any packets of one or more different priority levels are available.

[0029] **Figure 5** is a flow chart of an embodiment of adaptive queue scheduling utilizing a timer as a delay device. In this illustration, received packets are copied into a copied packets queue for scheduling **505**. The copied packets queue includes packets arranged in a number of sub-queues, each sub-queue representing one of a number of different priorities. In this example, three priority levels (P1 - high, P2 - medium, P3 low) are again provided, but any number and type of priority levels may be provided. To the degree that packets are available, a certain number of packets are scheduled from each priority level. In this weighted scheduling algorithm, the scheduling of the packets is weighted according to the priority levels of the packets. A driver **525** is sent three P1 packets **510**, two P2 packets **515**, and one P3 packet **520** (if this number of packets is available for each priority level) in each scheduling action.

[0030] After this set of packets has been scheduled and sent, there is a determination whether all of the sub-queues are empty **535**. If so, then there are no

packets available for scheduling, and the process continues with copying received packets into the copied packets queue **540**. There is a determination whether the P1 sub-queue remains empty **550**. If so, then there presently are no high priority packets available for copying. A timer is set **555**. The timer runs for a period of time, and during such period of time there are no attempts to copy received packets when it is determined that there are no high priority packets available. The timer providers a delay to allow the arrival of more high priority packets. While this example utilizes a timer to provide a delay period, this process can be provided by any delay process or mechanism that delays the copying of received packets. After setting the timer, the process then returns to sending packets from each of the priority queues **510-520**. If all of the priority queues are not empty **535**, there is then a determination whether the high priority queue P1 is empty **545**. If the P1 queue is not empty, then the process of scheduling packets from the priority queues continues **510-520**. If the P1 queue is empty, then there is a determination whether the timer is running (ON). If the timer is running, indicating that a delay period is currently active, then the process of scheduling packets from the priority queues continues **510-520**. If the timer is no longer ON, indicating that any delay period has ended or expired, then the process provides for copying received packets into the copied packets queue **540** and then making a determination whether the P1 sub-queue remains empty **550**.

[0031] Techniques described here may be used in many different environments. **Figure 6** is block diagram of an embodiment of an exemplary computer. Under an embodiment of the invention, a computer **600** comprises a bus **605** or other communication means for communicating information, and a processing means such as

one or more physical processors **610** (shown as **611**, **612** and continuing through **613**) coupled with the first bus **605** for processing information. Each of the physical processors may include multiple logical processors, and the logical processors may operate in parallel in the execution of drivers. Each processor may include an execution unit and logic for the operation of certain functions.

[0032] The computer **600** further comprises a random access memory (RAM) or other dynamic storage device as a main memory **615** for storing information and instructions to be executed by the processors **610**. RAM memory may include dynamic random access memory (DRAM) and static dynamic random access memory (SRAM). Main memory **615** also may be used for storing temporary variables or other intermediate information during execution of instructions by the processors **610**. Queues used in the scheduling of tasks may be implemented in the main memory **615**. The computer **600** also may comprise a read only memory (ROM) **620** and/or other static storage device for storing static information and instructions for the processor **610**.

[0033] A data storage device **625** may also be coupled to the bus **605** of the computer **600** for storing information and instructions. The data storage device **625** may include a magnetic disk or optical disc and its corresponding drive, flash memory or other nonvolatile memory, or other memory device. Such elements may be combined together or may be separate components, and utilize parts of other elements of the computer **600**.

[0034] The computer **600** may also be coupled via the bus **605** to a display device **630**, such as a liquid crystal display (LCD) or other display technology, for displaying information to an end user. In some environments, the display device may be a touch-screen that is also utilized as at least a part of an input device. In some

environments, display device **630** may be or may include an auditory device, such as a speaker for providing auditory information. An input device **640** may be coupled to the bus **605** for communicating information and/or command selections to the processor **610**. In various implementations, input device **640** may be a keyboard, a keypad, a touch-screen and stylus, a voice-activated system, or other input device, or combinations of such devices. Another type of user input device that may be included is a cursor control device **645**, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor **610** and for controlling cursor movement on display device **630**.

[0035] A communication device **650** may also be coupled to the bus **605**. Depending upon the particular implementation, the communication device **650** may include a transceiver, a wireless modem, a network interface card, or other interface device. The computer **600** may be linked to a network or to other devices using the communication device **650**, which may include links to the Internet, a local area network, or another environment.

[0036] In the description above, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some of these specific details. In other instances, well-known structures and devices are shown in block diagram form.

[0037] The present invention may include various processes. The processes of the present invention may be performed by hardware components or may be embodied in machine-executable instructions, which may be used to cause a general-purpose or

special-purpose processor or logic circuits programmed with the instructions to perform the processes. Alternatively, the processes may be performed by a combination of hardware and software.

[0038] Portions of the present invention may be provided as a computer program product, which may include a machine-readable medium having stored thereon instructions, which may be used to program a computer (or other electronic devices) to perform a process according to the present invention. The machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, CD-ROMs, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, magnet or optical cards, flash memory, or other type of media / machine-readable medium suitable for storing electronic instructions. Moreover, the present invention may also be downloaded as a computer program product, wherein the program may be transferred from a remote computer to a requesting computer by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem or network connection).

[0039] Many of the methods are described in their most basic form, but processes can be added to or deleted from any of the methods and information can be added or subtracted from any of the described messages without departing from the basic scope of the present invention. It will be apparent to those skilled in the art that many further modifications and adaptations can be made. The particular embodiments are not provided to limit the invention but to illustrate it. The scope of the present invention is not to be determined by the specific examples provided above but only by the claims below.

[0040] It should also be appreciated that reference throughout this specification to "one embodiment" or "an embodiment" means that a particular feature may be included in the practice of the invention. Similarly, it should be appreciated that in the foregoing description of exemplary embodiments of the invention, various features of the invention are sometimes grouped together in a single embodiment, figure, or description thereof for the purpose of streamlining the disclosure and aiding in the understanding of one or more of the various inventive aspects. This method of disclosure, however, is not to be interpreted as reflecting an intention that the claimed invention requires more features than are expressly recited in each claim. Rather, as the following claims reflect, inventive aspects lie in less than all features of a single foregoing disclosed embodiment. Thus, the claims are hereby expressly incorporated into this description, with each claim standing on its own as a separate embodiment of this invention.